



Docket No.: 62807-150

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of : Customer Number: 20277
Atsushi OTAKE : Confirmation Number: 2752
Serial No.: 10/715,121 : Group Art Unit: 2143
Filed: November 18, 2003 : Examiner: Unknown
For: PROGRAM CHANGING METHOD

TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT

Mail Stop CPD
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

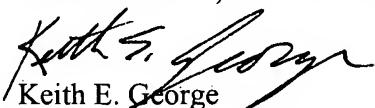
At the time the above application was filed, priority was claimed based on the following application:

Japanese Patent Application No. 2003-057936, filed March 5, 2003

A copy of the priority application listed above is enclosed.

Respectfully submitted,

MCDERMOTT, WILL & EMERY


Keith E. George
Registration No. 34,111

600 13th Street, N.W.
Washington, DC 20005-3096
(202) 756-8000 KEG:tlb
Facsimile: (202) 756-8087
Date: March 1, 2004

62807-150
OTAKE
November 18, 2003

日本国特許庁
JAPAN PATENT OFFICE

McDermott, Will & Emery

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: 2003年 3月 5日

出願番号
Application Number: 特願2003-057936

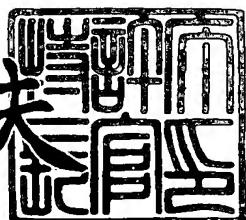
[ST. 10/C]: [JP2003-057936]

出願人
Applicant(s): 株式会社日立製作所

2003年10月28日

特許庁長官
Commissioner,
Japan Patent Office

今井康夫



【書類名】 特許願

【整理番号】 K02013801A

【あて先】 特許庁長官殿

【国際特許分類】 G06F 11/28

【発明者】

【住所又は居所】 神奈川県横浜市戸塚区戸塚町5030番地 株式会社日立製作所 ソフトウェア事業部内

【氏名】 大竹 厚

【特許出願人】

【識別番号】 000005108

【氏名又は名称】 株式会社日立製作所

【代理人】

【識別番号】 100075096

【弁理士】

【氏名又は名称】 作田 康夫

【手数料の表示】

【予納台帳番号】 013088

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 プログラム管理方法及びシステム並びにその処理プログラムを格納した記憶媒体

【特許請求の範囲】

【請求項 1】 所定URLへアクセス要求を送付して、エラーが発生したとき、WSDLファイルの変更によるエラーか否かを判定し、WSDLファイルの変更によるエラーの場合は、上記プログラムの変更箇所を決定し、所定手続きに基づいて、上記変更箇所を変更して、上記プログラムを実行するWebアクセス方法および装置並びにその処理プログラム。

【請求項 2】 前記アクセス要求送付のエラーの検出は、エラーが発生した場合に受け取るfaultメッセージにより起動され、faultメッセージの内容を解析し、その種類に応じて、送付したアクセス要求メッセージのエラーかどうかを判断するように構成したことを特徴とする請求項 1 記載のプログラム管理方法。

【請求項 3】 前記プログラムの変更箇所の検出は、請求項 2 記載のアクセス要求送付エラー検出の判断により起動され、エラーが生じたアクセス要求メッセージの解析を行い、その結果により改訂前WSDLファイルと改訂後WSDLファイルを比較することにより、発生したエラーがWSDLファイルの変更によるエラーかどうかを判断し、その結果によりプログラムの変更箇所を検出するように構成したことを特徴とする請求項 1 記載のプログラム管理方法。

【請求項 4】 前記プログラムの変更箇所の変更は、請求項 3 記載の変更箇所検出の判断により起動され、プログラムの変更箇所を変更するように構成したことを特徴とする請求項 1 記載のプログラム管理方法。

【請求項 5】 前記プログラム変更箇所変更後の実行は、請求項 4 記載の変更箇所変更の判断により起動され、変更したプログラムを実行するように構成したことを特徴とする請求項 1 記載のプログラム管理方法。

【請求項 6】 前記アクセス要求送付のエラー検出は、受け取ったfaultメッセージの種類が、送付したアクセス要求メッセージのエラーであることを解析するように構成したことを特徴とした請求項 2 記載のプログラム管理方法。

【請求項 7】 前記プログラムの変更箇所の検出は、エラーが発生したアクセス要

求メッセージをプログラムから取得し、そのメッセージがプログラム作成時に取得保存した改訂前WSDLファイルにしたがって作成されているかを解析し、その結果により変更箇所を検出するように構成したことを特徴とした請求項3記載のプログラム管理方法。

【請求項8】前記プログラムの変更箇所の検出は、送付アクセス要求メッセージの正否確認により、改訂前WSDLファイルから生成された送付アクセス要求メッセージであることを確認できた場合に、プログラム作成時に取得保存した改訂前WSDLファイルの取得条件から改訂後WSDLファイルの保存場所情報を取得するように構成したことを特徴とした請求項3記載のプログラム管理方法。

【請求項9】前記プログラムの変更箇所の検出は、送付アクセス要求メッセージの正否確認の判断結果により取得した改訂後WSDLファイルの保存場所情報から、改訂後WSDLファイルを取得するように構成したことを特徴とした請求項3記載のプログラム管理方法。

【請求項10】前記プログラムの変更箇所の検出は、送付アクセス要求メッセージの正否確認の判断結果により取得した改訂後WSDLファイルの保存場所情報から、改訂後WSDLファイルを取得し、改訂前WSDLファイルと改訂後WSDLファイルを比較し、WSDLファイルの変更の種類を解析するように構成したことを特徴とした請求項3記載のプログラム管理方法。

【請求項11】前記プログラムの変更箇所の検出は、送付アクセス要求メッセージの正否確認の判断結果により取得した改訂後WSDLファイルの保存場所情報から、改訂後WSDLファイルを取得保存し、WSDLファイル変更の種類の解析結果から、プログラムの変更箇所を検出するように構成したことを特徴とした請求項3記載のプログラム管理方法。

【請求項12】前記プログラムの変更箇所の変更は、請求項3記載のプログラム変更箇所の検出により、検出された変更内容に従い、プログラムの変更箇所の生成を行うように構成したことを特徴とする請求項4記載のプログラム管理方法。

【請求項13】前記プログラムの変更箇所変更後の実行は、請求項1.2記載の変更により生成したプログラムを実行するように構成したことを特徴とする請求項5記載のプログラム管理方法。

【発明の詳細な説明】**【0001】**

【発明の属する技術分野】 本発明は、プログラムの変更管理に関する。

【従来の技術】 近年、クライアント／サーバプログラムは分散オブジェクト技術を利用し、資源の有効利用、部品化が進んできている。インターネットを用い、ローカルなネットワークを超えた分散オブジェクト技術として、David A. Cappelli他著「Java Web Services」（出版社：O'Reilly & Associates, Inc., 2002.3）（JavaおよびJava関連の商標およびロゴは、米国もしくはその他のSun Microsystems, Inc. の商標または登録商標です）に記載されているWebサービスがある。本文献では、Webサービスの基礎技術として3つの技術を挙げており、SOAP(Simple Object Access Protocol)、WSDL(Web Service Description Language)、UDDI(Universal Description Discovery, and Integration)について記述している。

【0002】

本文献によれば、SOAPは構造化文書交換に使う封筒構造の定義であり、この定義にしたがって記述した構造化文書をSOAPメッセージと呼んでいる。Webサービスを用いたクライアント／サーバプログラムではこのSOAPメッセージを用いて通信を行う。このような技術は本文献 P.25～P53に開示されている。

次にWSDLとは、Webサービスのインターフェースを定義するための構造化文書フォーマットである。Webサービスを用いたクライアント／サーバプログラムでは、このWSDLフォーマットにしたがって記述したWSDLファイルをインターフェースとし、SOAPメッセージを用いて通信を行う。このような技術は本文献P.72～P95に開示されている。

最後に、UDDIとは、Webサービスに対するグローバルレジストリと標準仕様を提供しており、インターネットで共通に利用できるレジストリとして公開されている。したがって、Webサービスを用いたクライアント／サーバプログラムでは、サーバプログラムのWebサービスをこのUDDIレジストリに登録しておき、クライアントプログラムはこのUDDIレジストリに登録されているWebサービスを見つけることにより、サーバプログラムのWebサービスが利用できるようになる。このような技術は本文献 P.98～P139に開示されている。

【0003】**【非特許文献1】**

David A. Chappell他著「Java Web Services」（出版社：O'Reilly & Associates, Inc., 2002.3）

【0004】**【発明が解決しようとする課題】**

分散技術におけるアクセス要求送付先のインターフェース定義情報が変更されても、クライアントプログラムを変更し、作成する場合を考えてみる。保守作業者は、クライアントプログラムがエラーメッセージを受け付けた原因がインターフェース定義情報の変更によるものであることをわかっており、それをもとにクライアントプログラムを修正するものとする。

【0005】

上記のようなクライアントプログラムの保守作業に際しては、エラーを発生させた送付メッセージが、改定前のインターフェース定義情報にしたがって、生成されたメッセージかどうかを確認し、生成されたメッセージが正しい場合は、改訂後のインターフェース定義情報を取得し、改訂後のインターフェース定義情報にしたがったメッセージを送付するようにクライアントプログラムを修正する。

【0006】

しかしながら、従来の方式によるクライアントプログラムの生成は、インターフェース定義情報と雛型からクライアントプログラムを生成するものにすぎない。そのため、実行中のクライアントプログラムに発生したエラーがインターフェース定義情報の変更によるエラーであると原因がわかつても、保守作業者は、一旦クライアントプログラムを停止させ、改訂後のインターフェース定義情報を取得し、それにそったクライアントプログラムの雛型の作成を再度行い、インターフェース定義情報の変更箇所とプログラムの修正箇所の対応関係を判断しながら作業を行う手間のかかる作業となってしまう。

【0007】

本発明の目的は、実行中クライアントプログラムにおける、サーバプログラムの変更による修正と実行を動的に行うようにすることにある。

【0008】

【課題を解決するための手段】

前記課題を改善するため、クライアントプログラムが、サーバプログラムを提供している所定URLへアクセス要求を送付して、エラーが発生したとき、このエラーの発生原因を解析するためのプログラムを起動し、インターフェース定義情報の変更によるエラーか否かを判定し、インターフェース定義情報の変更によるエラーの場合は、上記クライアントプログラムの変更箇所を解析するプログラムを起動し、変更箇所の特定を行い、所定手続きに基づいて上記クライアントプログラムの上記変更箇所を変更して、上記クライアントプログラムを実行する。

【0009】

【発明の実施の形態】

以下、本発明の第一の実施例を図面に基づいて説明する。

図1は本実施例のシステム構成を示す。ネットワーク5に接続されたコンピュータ1はプログラム管理機能を備えたクライアントプログラムサーバであり、コンピュータ2はネットワークを介してクライアントプログラムサーバのプログラムを利用する一般ユーザである。ネットワークに接続されたコンピュータ3はwebサービスを提供するプログラムを格納したwebサービスサーバである。またコンピュータ4はwebサービスを登録管理するデータベースを格納したUDDIレジストリである。本実施例はサーバークライアント間の通信手段について特定するものではないが、インターネット上のHTTPプロトコルによる通信を前提として説明する。すなわち、クライアントプログラムサーバ1はHTTPサーバとしての機能を備え、一般ユーザ2はwebブラウザを介してクライアントプログラムサーバ1と通信するものとする。webサービスサーバ3はHTTPサーバとしての機能を備え、クライアントプログラムサーバ1とSOAPメッセージを用いて通信するものとする。また、UDDIレジストリ4はHTTPサーバとしての機能を備え、クライアントプログラムサーバ1とSOAPメッセージを用いて通信するものとする。従ってクライアントプログラムサーバ1は、ディスプレイ6とキーボード等のデータ入力装置7と、CPU8と、メモリ9と、改定前後のWSDLファイルを蓄積するデータベース10から構成され、メモリ9には、変更管理プログラム72が保持される。この変更

管理プログラム72には、開発環境70と実行環境71とが含まれる。実行環境71にはHTTPサーバプログラム24と、クライアントプログラム25と、クライアントプログラム実行プログラム26と、アクセス要求エラー検出プログラム27と、プログラムエラー発生個所特定プログラム28が含まれる。開発環境70には、エラー回避クライアントプログラム生成プログラム29が含まれる。また、一般ユーザ2は、ディスプレイ11と、入力装置12と、CPU13と、メモリ14から構成され、メモリ14にはwebブラウザプログラム31が保持される。webサービスサーバ3は、ディスプレイ15と、入力装置16と、CPU17と、メモリ18とデータベース30から構成され、メモリ18には、HTTPサーバプログラム32と、webサービスプログラム33が保持される。また、webサービスサーバ3は、改定前WSDLファイル36と改定後WSDLファイル37をデータベース30に保持している。UDDIレジストリ4は、ディスプレイ19と、入力装置20と、CPU21と、メモリ22と、データベース23から構成され、メモリ22には、HTTPサーバプログラム34と、UDDIプログラム35が保持される。また、UDDIレジストリ4は、データベース68に改訂後WSDLファイル保存場所情報51を保持している。

【0010】

図2は本実施例で実行されるクライアントプログラム25の構成を示す。図に示すように、制御機能38、業務処理機能39、表示機能40、アダプタプログラム41、SOAPクライアントプログラム42、WSDLスタブプログラム43から構成される。

一般ユーザクライアント2がクライアントプログラム25にリクエストを送付し、クライアントプログラム25がwebサービスサーバ3にアクセス要求を送付、処理結果を受け取り、一般ユーザクライアントに処理結果を表示する。

Webサービスサーバ3は住民基礎情報である住民の名前、住所、年齢等の情報を登録管理するサーバプログラムであり、アクセス要求を受け付けるメッセージとしてSOAPメッセージを用い、インターフェース定義情報の記述にはWSDLを用いる。WSDLを用いて記述した改訂前WSDLファイル36のmessageタグ部分を図3に示す。101はmessageタグであり、アクセス要求を受け付けるメソッド名の定義

としてname属性にsetJukiListを指定している。102はpartタグであり、setJukiListメソッドの引数の定義としてname属性にNameを指定し、この型定義としてtype属性にstringを指定している。103、104も同様にstring型で、Address変数とAge変数を指定している。

【0011】

図4に改訂後WSDLファイル37のmessageタグ部分を示す。改訂前WSDLファイル36に105のpartタグが付加され改訂されている。105はpartタグのname属性としてBirthdayを指定し、type属性にstringを指定している。

【0012】

以下、webサービスサーバ3の管理者が住民基礎情報の内容充実化をはかるために、登録関数の引数一つを増やし、生年月日を追加することにしたものとして、説明を行う。したがって、UDDIレジストリに登録する自サービスのインターフェースである改訂前WSDLファイル36の保存場所情報を改訂後WSDLファイル37の保存場所情報に入れ替えを行った。この時の改定後WSDLファイル37のmessageタグ部分は図4に示す。

【0013】

クライアントプログラム2.5の表示機能40は、図5に示すHTMLを記述した改訂前のファイルであり、201はhtmlタグであり、202はheadタグ、203はtitleタグであり「住民基礎情報入力」をwebブラウザタイトルに表示するように定義している。204はbodyタグであり、ここからwebブラウザに表示される内容であることを示している。したがって205の文字列「住民基礎情報入力」の記述でwebブラウザに文字列「住民基礎情報入力」が表示される。206は一般ユーザクライアント2が入力するFORMタグであり、httpリクエストのPOSTメソッド実行を定義している。また、このときのアクションタイプとして、SETJUKIを定義している。

【0014】

207から209はこのFORMタグで送信される内容を定義している。テキスト入力項目として207のName、208のAddress、209のAgeと、このテキスト入力項目207、208、209の情報を送信するためのsubmitタグ210と、

テキスト入力項目の内容を消去するresetタグ211から構成される。この改訂前JSPファイルが一般ユーザクライアント2のwebブラウザに表示される画面を図7に示す。

【0015】

改訂後のJSPファイルの内容を図6に示す。改訂前JSPファイルにテキスト入力項目212のBirthdayが付加され改訂されている。この改訂後JSPファイルが一般ユーザクライアント2のwebブラウザに表示される画面を図8に示す。

【0016】

次にテキスト入力項目とWSDLファイルのpartタグとの関係を説明する。図5の改訂前JSPファイルのテキスト入力項目は、図3に示す改定前WSDLファイル36のmessageタグの子要素であるpartタグで記述された引数と同一の項目である。つまり、partタグで記述された102のName、103のAddress、104のAgeが図5のJSPファイルのテキスト入力項目である207のName、208のAddress、209のAgeとなる。

次に、このテキスト入力項目がクライアントプログラム25で処理される手順を説明する。

【0017】

図2において一般ユーザクライアント2がwebブラウザプログラム31から入力したテキスト入力項目の内容は25_1のHTTPリクエストでクライアントプログラム25へ送られ、制御機能38がリクエストを受け付ける。このリクエストは25_2で業務処理機能39へ処理要求が行われる。25_3で業務処理機能39はアダプタプログラム41へ、リクエストを引数として処理を依頼する。アダプタプログラム41はリクエストのテキスト入力項目の内容を取り出し、SOAPクライアントプログラム42のメソッドを実行し、WSDLスタブプログラム43を使用し、webサービスサーバ3にアクセス要求を送付し、webサービスを利用する。webサービスサーバ3はアクセス要求を受け付けた場合、25_4で結果として戻り値を返却する。返却された戻り値は、WSDLスタブ43、SOAPクライアントプログラム42、アダプタプログラム41を通じて業務処理機能39に格納される。25_6で制御機能38は業務処理機能39の処理終了の通知を受けて、2

5_7の表示機能4_0に出力要求を行う。25_8で表示機能4_0では業務処理機能3_9に格納されている戻り値を取得し、25_9で結果を受け取る。25_9で受け取った結果をレスポンスとして一般ユーザクライアント2のwebブラウザプログラム3_1に表示する。

【0018】

次に、クライアントプログラム2_5で動作する各プログラムについて説明する。

WSDLスタブプログラム4_3は図3の改定前WSDLファイル3_6から生成される。図9に改定前WSDLファイル3_6から生成されたWSDLスタブプログラムのソースコードを示す。201_0はWSDLスタブプログラムのパッケージ宣言である。ここでは、パッケージとしてjp.co.Ohtake.wsdlというパッケージ名称とし各クラスはこのパッケージに生成されることを意味する。202_0はJukiList_Serviceのインターフェース宣言であり、メソッドとして203_0のsetJukiListを持つことを宣言している。また、この203_0では、このメソッドの引数としてString型のName、Address、Ageを持ち、戻り値としてboolean型を返すことを定義している。204_0は202_0の定義の終わりを表している。

【0019】

205_0は201_0と同様にパッケージ宣言である。206_0はregistry.Registryクラスのインポート宣言をしている。207_0ではregistry.RegistryExceptionクラスのインポート宣言をしている。208_0から215_0ではJukiList_ServiceHelperクラスを定義しており、209_0と212_0にこのクラスが持つメソッドを定義している。210_0では209_0で宣言されたメソッドbind()の戻り値を定義している。211_0は209_0の定義の終わりである。213_0では212_0で宣言されたメソッドbind(String url)の戻り値を定義している。

【0020】

図1_0は図4に示す改訂後WSDLファイル3_7から生成されるWSDLスタブプログラムのソースコードである。図9に示すソースコードとの違いは、203_0に示すsetJukiListメソッドの引数にString型Birthdayが追加されたことである。これを図1_0の216_0に示す。

【0021】

次に、改訂前WSDLファイル36から生成された図9のWSDLスタブプログラムを利用するSOAPクライアントプログラム42のソースコードを図11に示す。3010はパッケージ宣言である。3020から3070でInvoke_jukiListクラスを定義している。3030はこのInvoke_jukiListクラスのメソッドの宣言であり、引数としてString型のName、Address、Ageと戻り値としてboolean型を定義している。3040は2090から2110で定義されたJukiList_ServiceHelperクラスのメソッドbind()を用いてJukiList_Serviceクラスのインスタンスを得ている。次に3050では、JukiList_Serviceクラスのインスタンスを用いて2030で定義されたメソッドsetJukiListを使用し、結果を得ている。

【0022】

図12に図11のSOAPクライアントプログラムを利用するアダプタプログラム41のソースコードを示す。アダプタプログラム41は4010でjp.co.Ohtake.adapterパッケージ宣言をしている。4020ではjp.co.Ohtake.wsdlパッケージ全てをインポートし、4030ではjavax.servlet.httpパッケージ全てをインポートしている。4040から4150では、adp_Invoke_jukiListクラスの定義を行っている。4050から4140ではこのクラスのメソッド定義をしている。このメソッドadp_setjukiは戻り値がBoolean型で引数がHttpServletRequest型requestである。

【0023】

4060では戻り値用の変数adp_rtnの初期化をfalseで行っている。次に4070から4090ではrequest変数が保持している値をそれぞれ取り出し、adp_name、adp_address、adp_ageへと代入している。4100では3020で定義されたクラスのインスタンスを生成している。4120ではこのインスタンスでメソッドsetjukiを実行し、結果をadp_rtnに代入している。

【0024】

図13は図10に示す改訂後WSDLファイル37から生成されるWSDLスタブプログラムのソースコードを利用するSOAPクライアントプログラムのソースコードである。図11との違いは、3080のメソッドの引数にString型のBirthdayが増

加している事と、3090のメソッドの引数にString型のBirthdayが増加している事である。

【0025】

図14は図13に示すSOAPクライアントプログラムを利用するためのアダプタプログラムのソースコードである。図12との違いは、4160と4170である。4160では、String型の変数adp_birthdayにrequest変数から値を取り出し、代入している。4170では、実行するメソッドsetjukiの引数adp_birthdayが増加している。

次に、webサービスサーバ3のwebサービスプログラム33のインターフェースが、改訂前WSDLファイル36から改訂後WSDLファイル37へ変更された場合における変更管理プログラム72の処理概要を図15に示す。

【0026】

Faultメッセージ44を、クライアントプログラム25が受け取り、アクセス要求エラーの検出プログラム27が実行され、プログラムエラー発生個所特定プログラム28が実行され、エラー回避クライアントプログラム生成プログラム29が実行され、クライアントプログラム実行プログラム26が実行される。

【0027】

クライアントプログラム25が、改訂後webサービスプログラム33を有するwebサービスサーバ3にアクセス要求を送付した結果、Faultメッセージ44を取得した時、クライアントプログラム25では、WSDLスタブプログラム43がFaultメッセージ44を取得し、SOAPクライアント42へ返信する。SOAPクライアント42はアダプタプログラム41へFaultメッセージ44を返信する。アダプタプログラム41がFaultメッセージ44を受け取ると、Faultメッセージ取得通知プログラム45を実行する。Faultメッセージ取得通知プログラム45を実行したアダプタプログラム41は完全停止フラグまたは、一時停止フラグに基づき処理結果を業務処理機能39に返却する。アダプタプログラム41から結果フラグを受け取った業務処理機能39は処理終了を制御機能38に伝える。処理終了を伝えられた制御機能38は表示機能40に出力要求を行う。出力要求を受けた表示機能40は業務処理機能39の処理結果を取得し、一般ユーザクライアント2

のwebブラウザプログラムにレスポンスを返す。

【0028】

図15に示した処理概要の詳細を以下に示す。

まず、アクセス要求エラー検出プログラム27の手順を図16に示す。アクセス要求エラー検出プログラム27は、Faultメッセージ取得通知プログラム45、Faultメッセージ解析プログラム46から構成される。

Faultメッセージ取得通知プログラム45の処理フローを図19に示す。Faultメッセージ取得通知プログラム45はFaultメッセージを受け取る（300）。次にFaultメッセージ解析プログラム46を実行し、Faultメッセージ44を送る（301）。Faultメッセージ解析プログラム46の戻り値を判定する（302）。Faultメッセージ解析プログラム46の戻り値がClientフラグの場合、アダプタプログラム41へ一時停止フラグを返信し（303）、Clientフラグ以外の場合、完全停止フラグをアダプタプログラム41へ返信する（304）。

Faultメッセージ解析プログラム46の処理フローを図20に示す。Faultメッセージ解析プログラム46は受け取ったFaultメッセージ44を解析する（400）。このメッセージに記述されている全てのタグについて以下の判定を行う（401）。

全てのタグの中からfaultCodeタグを見つけ出し（402）、このタグの値がClientエラーを表すエラーコードかどうかを判断する（403）。Clientエラーを表すエラーコードの場合、既存メッセージ確認プログラム47を実行する（404）。既存メッセージ確認プログラム47の戻り値がClientフラグの場合（405）、ClientフラグをFaultメッセージ取得通知プログラム45に返信し（406）、Clientフラグ以外の場合、Clientフラグ以外を返信する（407）。403でClientエラーを表すエラーコードでない場合、Faultメッセージ取得通知プログラム45へClientフラグ以外を返信する（408）。

プログラムエラー発生個所検出プログラム28の手順を図17に示す。プログラムエラー発生個所検出プログラム28は、既存メッセージ確認プログラム47、UDDI検索プログラム48、WSDLファイル取得プログラム52、WSDLファイル比較プログラム53から構成される。

【0029】

既存メッセージ確認プログラム47の処理フローを図21に示す。既存メッセージ確認プログラム47は、webサービスサーバ3にアクセス要求を送付したりクエストメッセージ49を受け取る（500）。次に、改訂前WSDLファイル36をデータベース10から取得し（501）、改訂前WSDLファイル36から確認用リクエストメッセージ67を生成する（502）。リクエストメッセージ49の全てのタグについて（503）、生成した確認用リクエストメッセージ67のタグと同一かどうか判定する（504）。

改訂前WSDLファイル36から生成されたメッセージである場合、UDDI検索プログラム48を実行する（505）。改訂前のWSDLファイル36から生成されたメッセージではない場合、改訂前のWSDLファイル36をWSDLスタブ生成プログラム54へ送る（509）。505で実行したUDDI検索プログラム48の戻り値を判定する（506）。戻り値がClientフラグの場合、Faultメッセージ解析プログラム46へClientフラグを返信し（507）、Clientフラグ以外の場合、Clientフラグ以外を返信する（508）。

UDDI検索プログラム48の処理フローを図22に示す。UDDI検索プログラム48は改訂前webサービス取得条件50をデータベース10から取得し（600）、この条件に基づき、UDDIレジストリ4でwebサービスの検索を行う。全てのwebサービスについて検索し（601）、改訂前webサービス取得条件50に一致するwebサービスを見つけることができた場合（602）、WSDLファイル保存場所情報が改訂されているか判定する（603）。

【0030】

WSDLファイル保存場所情報が改訂されている場合、UDDIレジストリ4に登録されている改訂後WSDLファイル保存場所情報49を取得し（604）、WSDLファイル取得プログラム52へ送る。WSDLファイル取得プログラム52の戻り値を判定し（605）、Clientフラグの場合、既存メッセージ確認プログラム47へClientフラグを返信し（606）、Clientフラグ以外の場合、Clientフラグ以外を返信する（607）。603でWSDLファイル保存場所情報が改訂されていない場合、既存メッセージ確認プログラム47へWSDL改訂なしエラーを返信する（608）。

）。602で改訂前webサービス取得条件50に一致するwebサービスを見つけることができなかった場合、既存メッセージ確認プログラム47へwebサービス未検出エラーを返信する（609）。

【0031】

WSDLファイル取得プログラム52の処理フローを図23に示す。WSDLファイル取得プログラム52は、改訂後WSDLファイル保存場所情報51を受け取り（700）、改訂後WSDLファイル保存場所情報51にしたがい、改訂後WSDLファイル37を取得する（701）。次に、改訂後WSDLファイル37が取得できたかどうかを判定し（702）、取得できた場合、取得した改訂後WSDLファイル37をWSDLファイル比較プログラム53へ送り、実行する（703）。実行したWSDLファイル比較プログラム53の戻り値について判定し（704）、戻り値がpartタグ変更である場合、UDDI検索プログラム48へClientフラグを返信し（705）、partタグ変更以外の場合、Clientフラグ以外を返信する（706）。702で改訂後WSDLファイル37を取得できなかった場合、UDDI検索プログラム48へ、取得失敗エラーを返す（707）。

【0032】

WSDLファイル比較プログラム53の処理フローを図24に示す。初めに、WSDLファイル比較プログラム53は改定後WSDLファイル37を取得し（800）、次に改定前WSDLファイル36を取得する（801）。

取得した改定前後のWSDLファイルの比較を行い（802）、抽出された相違部分がWSDLファイル中のmessageタグの子要素であるpartタグの増減であれば、WSDLスタブ生成プログラム54を実行し、改定後のWSDLファイル37を送る（803）。その後、WSDLファイル取得プログラム52へpartタグ変更を返信する（804）。802での判定がpartタグ増減以外の変更の場合、partタグ以外変更エラーをWSDLファイル取得プログラム52へ返信する（805）。

【0033】

図18に示すエラー回避クライアントプログラム生成プログラム29の手順を以下に示す。エラー回避クライアントプログラム生成プログラム29は、WSDLスタブ生成プログラム54、SOAPクライアント生成プログラム56、アダプタ生成

プログラム58、JSP生成プログラム60、クライアントプログラム生成プログラム62から構成される。

WSDLスタブ生成プログラム54の処理フローを図25に示す。WSDLスタブ生成プログラム54は改訂後WSDLファイル37を受け取り（900）、WSDLスタブソース55を生成するコマンドを実行する（901）。このWSDLスタブソース55をSOAPクライアント生成プログラム56へ送り実行する（902）。次に、JSP生成プログラム60へ改訂後WSDLファイル37を送り、実行する（903）。

【0034】

次に、SOAPクライアント生成プログラム56の処理フローを図26に示す。SOAPクライアント生成プログラム56はWSDLスタブソース55を受け取る（1000）。次に、WSDLスタブソース55を用いて、SOAPクライアントプログラムソース57を生成する（1001）。SOAPクライアントプログラムソース57の生成の詳細は1002から1007で行われ、生成結果を図13に示す。生成したSOAPクライアントプログラムソース57をアダプタ生成プログラム58へ送る（1008）。

【0035】

次に、アダプタ生成プログラム58の処理フローを図27に示す。アダプタ生成プログラム58は受け取ったSOAPクライアントプログラムソース57から（1100）、アダプタプログラムソース59を生成する（1101）。アダプタプログラムソース59の生成は1102から1108で行われ、生成した結果を図14に示す。次に図25の903で実行されたJSP生成プログラム60の処理が終了しているかどうかを判断し（1109）、アダプタプログラムソース59をクライアントプログラム生成プログラム62へ送り実行する（1110）。

【0036】

JSP生成プログラム60の処理フローを図28に示す。JSP生成プログラム60は改訂後WSDLファイル37を受け取る（1200）。この改訂後WSDLファイル37のmessageタグの引数部分を入力項目とするJSPファイル61を生成する（1201）。1202から1204にJSPファイル生成の詳細を示す。生成されたJSPファイル61の内容を図6に示す。次に、生成したJSPファイル61をクライア

ントプログラム生成プログラム62へ送り、実行する（1205）。

【0037】

クライアントプログラム生成プログラム62の処理を以下に説明する。クライアントプログラム生成プログラム62は、WSDLスタブソース55とSOAPクライアントプログラムソース57とアダプタソース59とJSPファイル61を受け取る。次に受け取ったWSDLスタブソース55とSOAPクライアントプログラムソース57とアダプタソース59をjavacコマンドによりコンパイルし、classファイルを生成する。生成された各classファイルをjarコマンドによりjarファイル形式のファイルJARファイル68にする。次に、JARファイル68とJSPファイル61をjarコマンドの-warオプションにてwarファイル形式のファイルWARファイル73にする。最後にこのWARファイル73をクライアントプログラム実行プログラム26へ送り、クライアントプログラム実行プログラム26を実行する。

【0038】

クライアントプログラム実行プログラム26は、改定前のWSDLファイル36に基づいたWARファイルを使用したクライアントプログラム25を停止し、クライアントプログラム生成プログラム62から受け取ったWARファイル73を使用したクライアントプログラムを配置し、実行する。

【0039】

【発明の効果】

本発明によれば、実行中のクライアントプログラムについて、インターフェースの変更による対応変更管理が動的に行えるため、サービス停止期間を短縮することが可能となる。

【図面の簡単な説明】

【図1】本発明を適用したシステム構成を示す図である。

【図2】本発明に適用されるクライアントプログラムの構成を示す図である。

【図3】本発明に適用される改定前WSDLファイルのメッセージ定義を示す図である。

【図4】本発明に適用される改定後WSDLファイルのメッセージ定義を示す図である。

【図5】本発明に適用されるクライアントプログラムの改訂前の表示機能を示す図である。

【図6】本発明に適用されるクライアントプログラムの改訂後の表示機能を示す図である。

【図7】本発明に適用される改訂前の一般ユーザクライアントに表示される画面を示す図である。

【図8】本発明に適用される改訂後の一般ユーザクライアントに表示される画面を示す図である。

【図9】本発明に適用される改訂前のWSDLスタブプログラムコードを示す図である。

【図10】本発明に適用される改訂後のWSDLスタブプログラムコードを示す図である。

【図11】本発明に適用される改訂前のSOAPクライアントプログラムコードを示す図である。

【図12】本発明に適用される改訂前のアダプタプログラムコードを示す図である。

【図13】本発明に適用される改訂後のSOAPクライアントプログラムコードを示す図である。

【図14】本発明に適用される改訂後のアダプタプログラムコードを示す図である。

【図15】本発明の実行形態の概略を示す図である。

【図16】アクセス要求エラーを検出し、Faultメッセージの解析ステップを示す図である。

【図17】改定後WSDLファイルを取得し、プログラムエラー発生個所検出ステップを示す図である。

【図18】プログラムエラー発生個所検出ステップで検出された変更部分について、クライアントプログラムを生成するエラー回避クライアントプログラムの生成巣ステップを示す図である。

【図19】本発明に適用されるFaultメッセージ取得通知プログラムの処理フロ

ーを示す図である。

【図20】本発明に適用されるFaultメッセージ解析プログラムの処理フローを示す図である。

【図21】本発明に適用される既存メッセージ確認プログラムの処理フローを示す図である。

【図22】本発明に適用されるUDDI検索プログラムの処理フローを示す図である。

【図23】本発明に適用されるWSDLファイル取得プログラムの処理フローを示す図である。

【図24】本発明に適用されるWSDLファイル比較プログラムの処理フローを示す図である。

【図25】本発明に適用されるWSDLスタブ生成処理プログラムの処理フローを示す図である。

【図26】本発明に適用されるSOAPクライアント生成プログラムの処理フローを示す図である。

【図27】本発明に適用されるアダプタ生成プログラムの処理フローを示す図である。

【図28】本発明に適用されるJSP生成プログラムの処理フローを示す図である。

【符号の説明】

1 … クライアントプログラム変更管理サーバ

2 … 一般ユーザクライアント

3 … webサービスサーバ

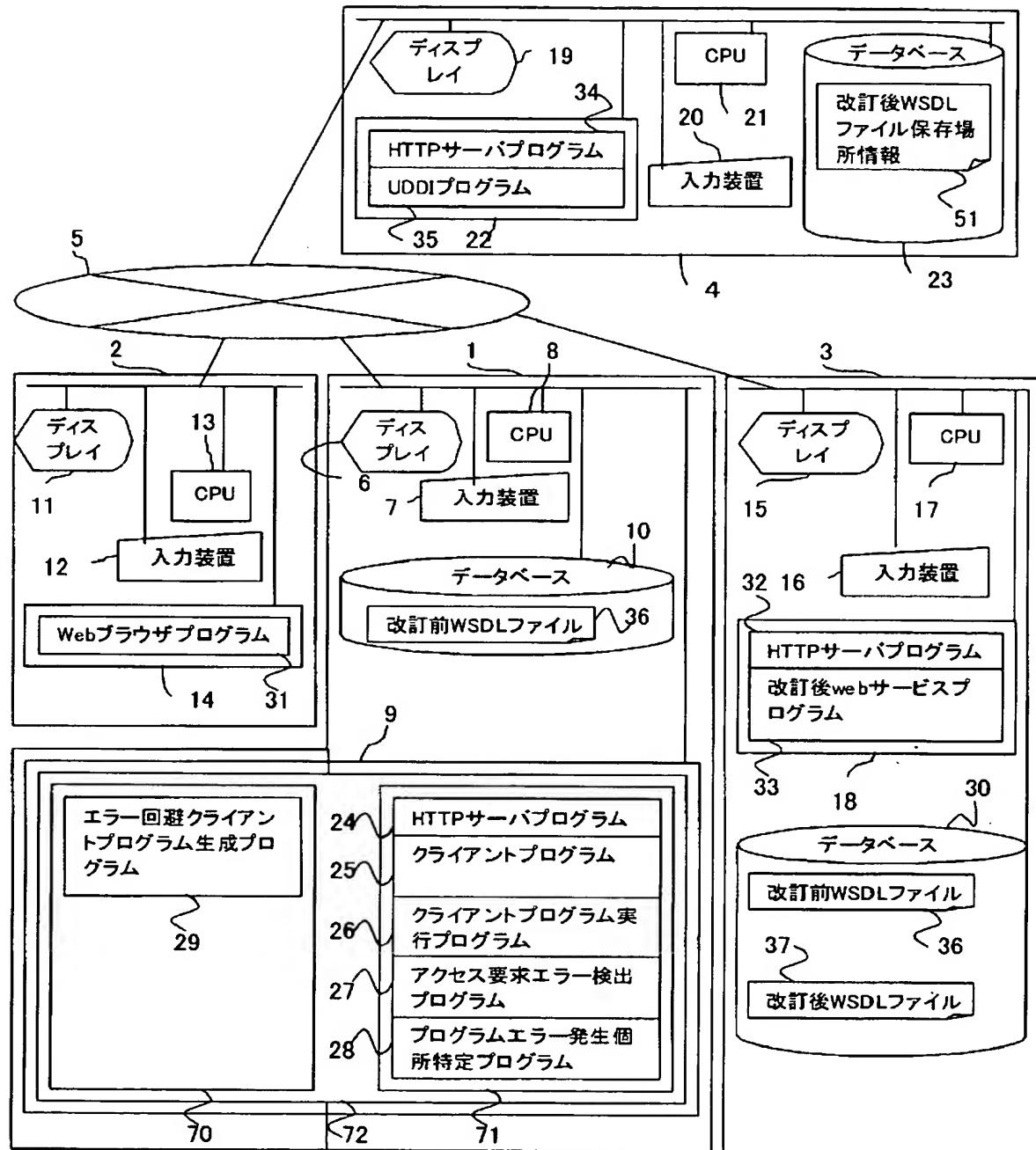
4 … UDDIレジストリサーバ

5 … ネットワーク

【書類名】 図面

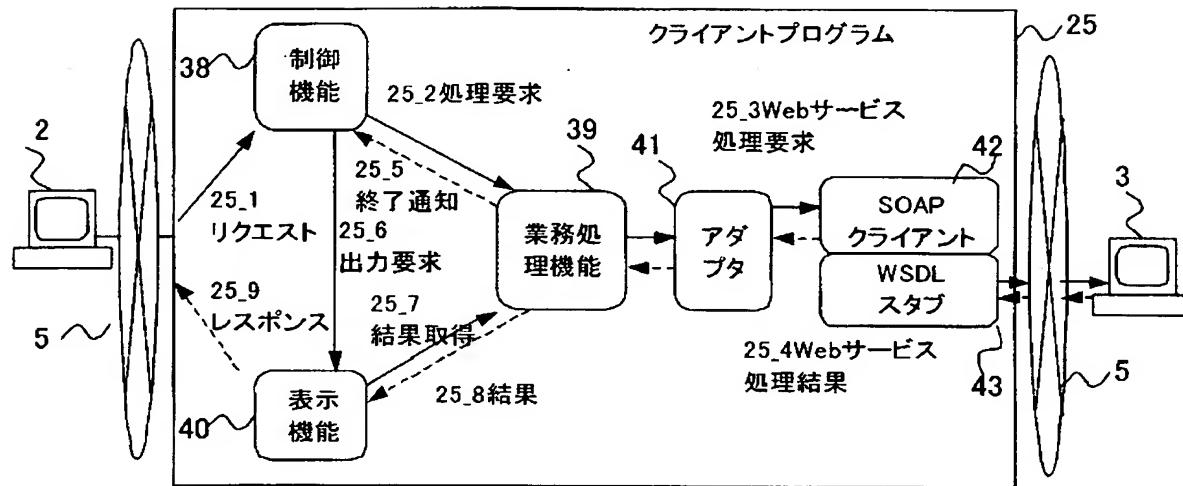
【図 1】

図 1



【図2】

図2



【図3】

図3

```

<message name="setJukiList"> 101
  < part name="Name" type="xsd:string"/> 102
  < part name="Address" type="xsd:string"/> 103
  < part name="Age" type="xsd:string"/>
</message> 104
:

```

【図4】

図4

```

<message name="setJukiList"> 101
  < part name="Name" type="xsd:string"/> 102
  < part name="Address" type="xsd:string"/> 103
  < part name="Age" type="xsd:string"/> 104
  < part name="Birthday" type="xsd:string"/>
</message> 105
:

```

【図5】

図5

```

<html> 201
<head> 202
<title>住民基礎情報入力</title> 203
</head>
<body> 204
<center>住民基礎情報入力 205
<form method="POST" action="SETJUKI"> 206
  <p>Name<input type="text" name="Name" size="100"></p> 207
  <p>Address<input type="text" name="Address" size="100"></p> 208
  <p>Age<input type="text" name="Age" size="100"></p> 209
  <p><input type="submit" value="送信" name="B1"> 210
    <input type="reset" value="リセット" name="B2"></p> 211
</form>
</center>
</body>
</html>

```

【図6】

図6

```

<html> 201
<head> 202
<title>住民基礎情報入力</title> 203
</head>
<body> 204
<center>住民基礎情報入力 205
<form method="POST" action="SETJUKI"> 206
  <p>Name<input type="text" name="Name" size="100"></p> 207
  <p>Address<input type="text" name="Address" size="100"></p> 208
  <p>Age<input type="text" name="Age" size="100"></p> 209
  <p>Birthday<input type="text" name="Birthday" size="100"></p> 212
  <p><input type="submit" value="送信" name="B1"> 210
    <input type="reset" value="リセット" name="B2"></p> 211
</form>
</center>
</body>
</html>

```

【図7】

図7

住民基礎情報入力 205

Name 207

Address 208

Age 209

210 211

【図8】

図8

住民基礎情報入力 205

Name 207

Address 208

Age 209

Birthday 212

210 211

【図 9】

図 9

```
2010 package jp.co.Ohtake.wsdl;  
  
2020 public interface JukiList_Service {  
2030   boolean setJukiList(String Name, String Address, String Age);  
2040 }  
  
2050 package jp.co.Ohtake.wsdl;  
  
2060 import registry.Registry;  
2070 import registry.RegistryException;  
  
2080 public class JukiList_ServiceHelper{  
2090   public static JukiList_Service bind() throws RegistryException{  
2100     return bind("http://www.ohtake_service/wsdl/JukiListService.wsdl");  
2110   }  
2120   public static JukiList_Service bind(String url) throws RegistryException{  
2130     return (JukiList_Service)Registry.bind(url, JukiList_Service.class);  
2140   }  
2150 }
```

【図10】

図10

```
2010 package jp.co.Ohtake.wsdl;

2020 public interface JukiList_Service {
2160   boolean setJukiList(String Name, String Address, String Age, String Birthday);
2040 }

2050 package jp.co.Ohtake.wsdl;

2060 import registry.Registry;
2070 import registry.RegistryException;

2080 public class JukiList_ServiceHelper{
2090   public static JukiList_Service bind() throws RegistryException{
2100     return bind("http://www.ohtake_service/wsdl/JukiListService.wsdl");
2110   }
2120   public static JukiList_Service bind(String url) throws RegistryException{
2130     return (JukiList_Service)Registry.bind(url, JukiList_Service.class);
2140   }
2150 }
```

【図11】

図11

```
3010 package jp.co.Ohtake.wsdl;

3020 public class Invoke_jukiList{
3030   public boolean setjuki(String Name, String Address, String Age) throws
Exception{
3040     JukiList_Service jukiListService = JukiList_ServiceHelper.bind();
3050     boolean rtn = jukiListService.setJukiList( Name, Address, Age );
3060   }
3070 }
```

【図1-2】

図1-2

```
4010 package jp.co.Ohtake.adapter;  
  
4020 import jp.co.Ohtake.wsdl.*;  
4030 import javax.servlet.http.*;  
  
4040 public class adp_Invoke_jukiList{  
4050     public Boolean adp_setjuki( HttpServletRequest request )throws Exception{  
4060         boolean adp_rtn = false;  
4070         String adp_name=request.getAttribute( "Name" );  
4080         String adp_address=request.getAttribute( "Address" );  
4090         String adp_age=request.getAttribute( "Age" );  
  
4100         Invoke_jukiList soapclient = new Invoke_jukiList();  
4120         adp_rtn = soapclient.setjuki(adp_name, adp_address, adp_age );  
4130         return adp_rtn;  
4140     }  
4150 }
```

【図1-3】

図1-3

```
3010 package jp.co.Ohtake.wsdl.  
  
3020 public class Invoke_jukiList{  
3080     public boolean setjuki(String Name,String Address,String Age,String  
Birthday ) throws Exception{  
3040         JukiList_Service jukiListService = JukiList_ServiceHelper.bind();  
3090         boolean rtn = jukiListService.setJukiList( Name, Address, Age, Birthday );  
3060     }  
3070 }
```

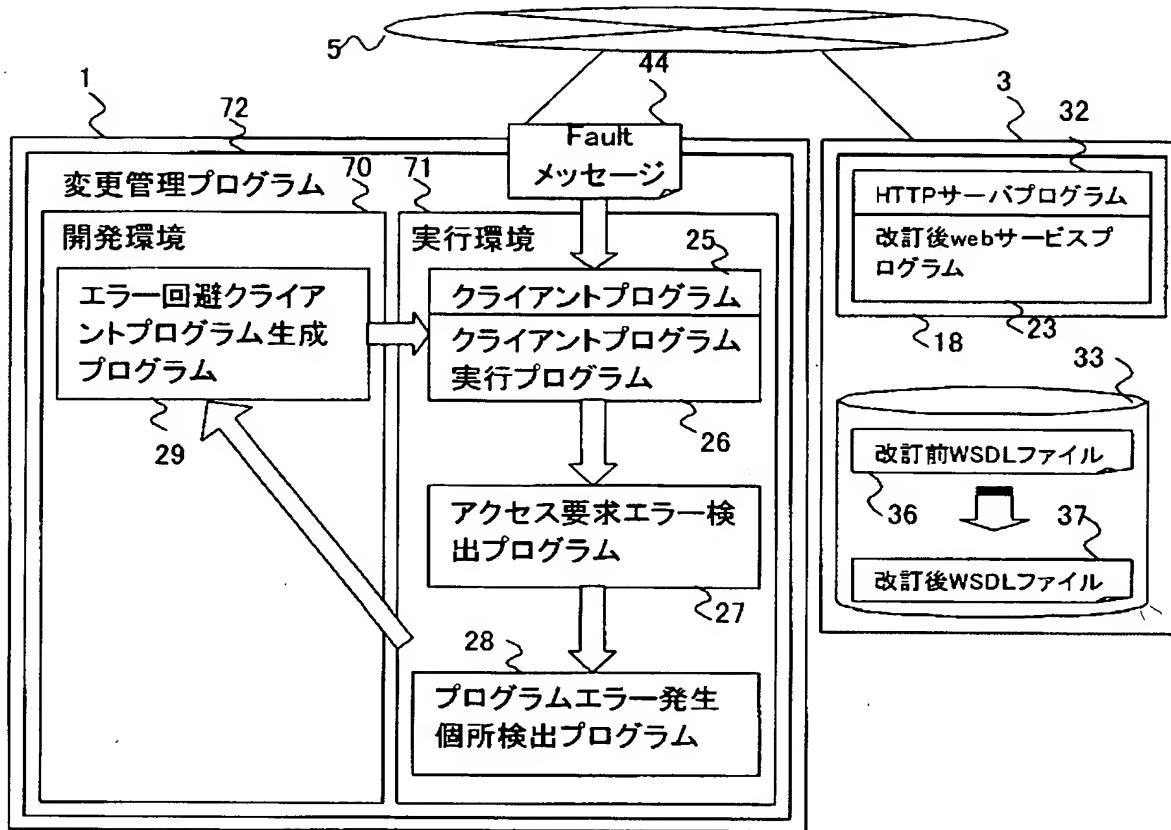
【図 1 4】

図 1 4

```
4010 package jp.co.Ohtake.adapter;  
  
4020 import jp.co.Ohtake.wsdl.*;  
4030 import javax.servlet.http.*;  
  
4040 public class adp_Invoke_jukiList{  
4050     public Boolean adp_setjuki( HttpServletRequest request )throws Exception{  
4060         boolean adp_rtn = false;  
4070         String adp_name=request.getAttribute( "Name" );  
4080         String adp_address=request.getAttribute( "Address" );  
4090         String adp_age=request.getAttribute( "Age" );  
4100         String adp_birthday=request.getAttribute( "Birthday" );  
  
4110         Invoke_jukiList soapclient = new Invoke_jukiList();  
4120         adp_rtn=soapclient.setjuki(adp_name,adp_address,adp_age,  
adp_birthday);  
4140     }  
4150 }
```

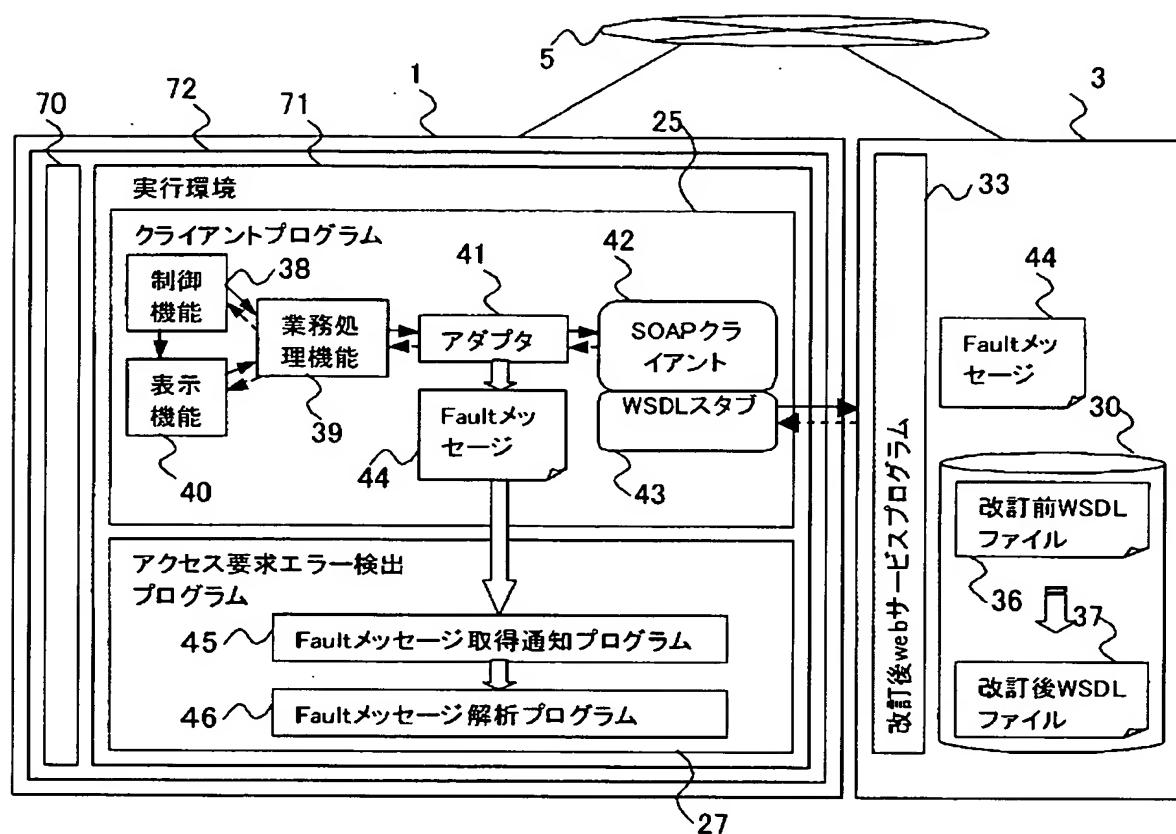
【図15】

図15



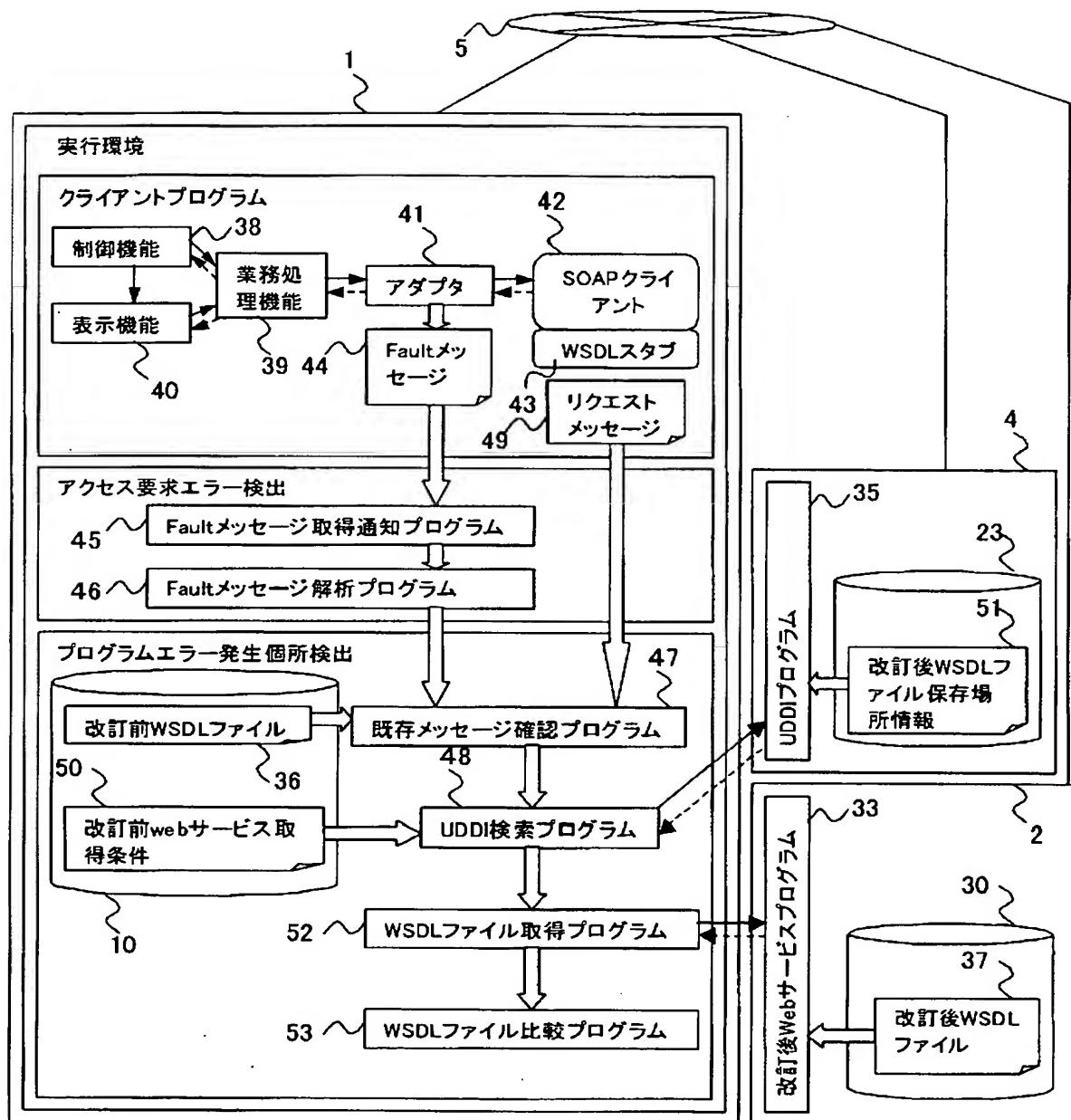
【図16】

図16



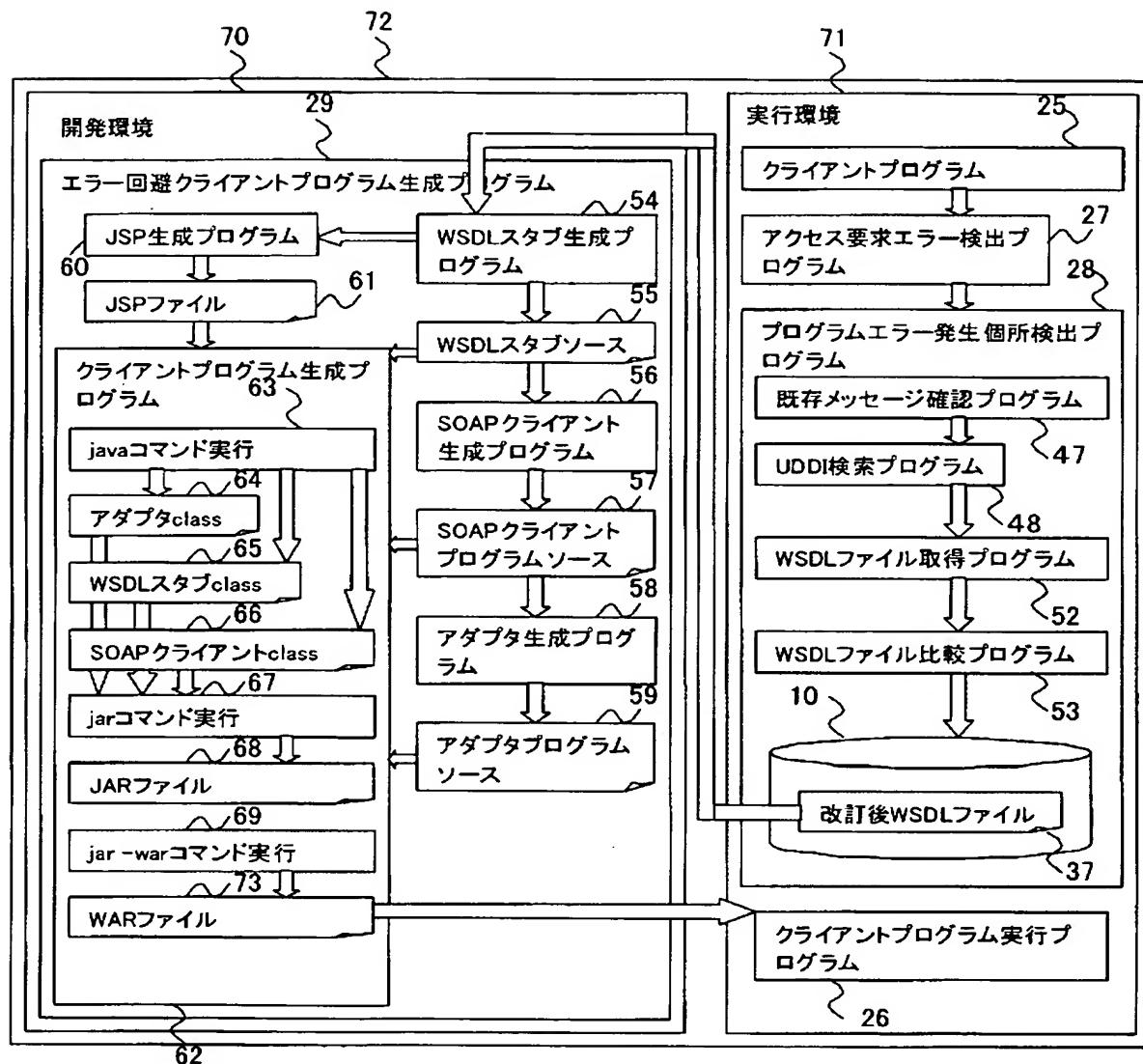
【図 17】

図 17



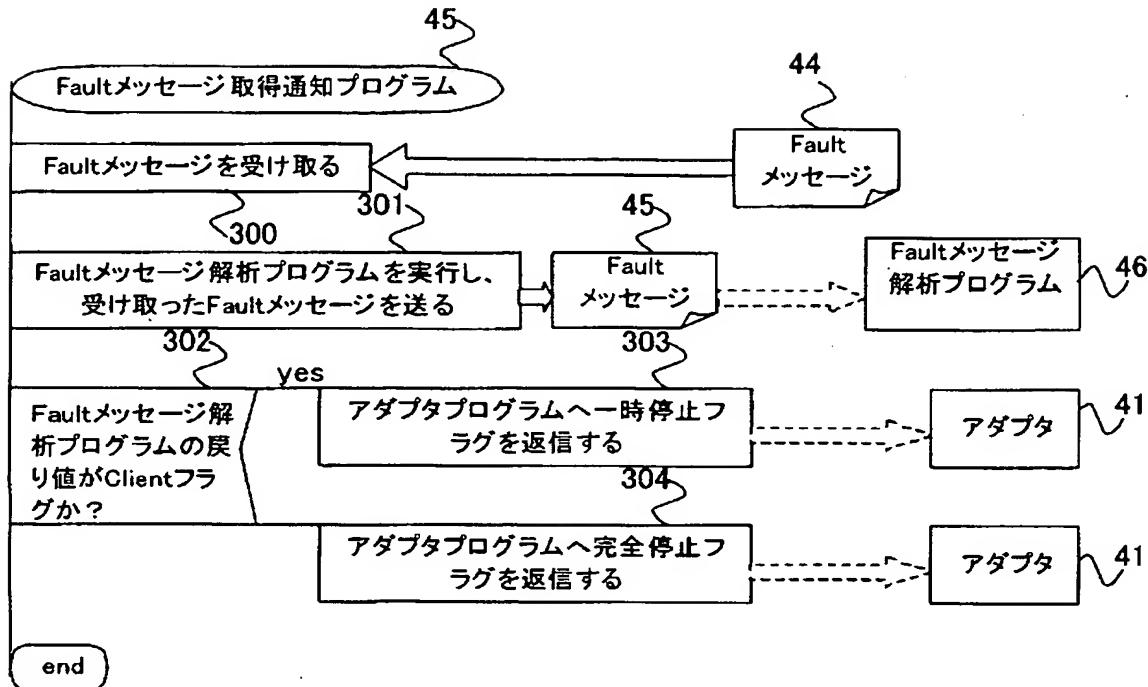
【図18】

図18



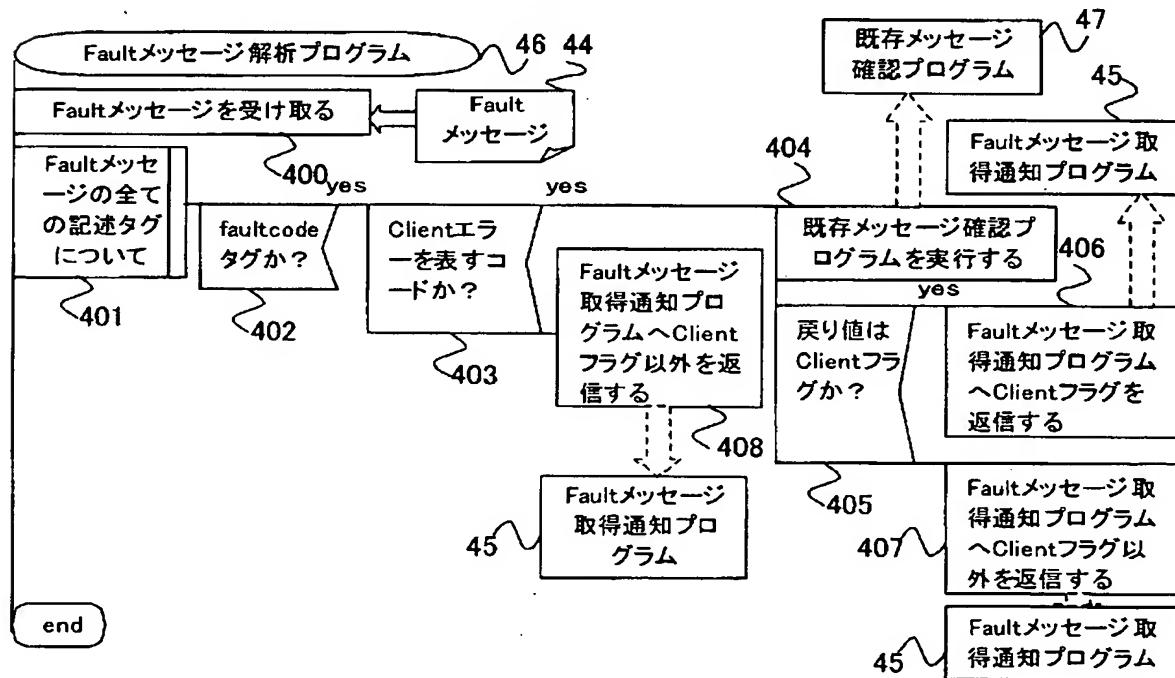
【図 19】

図 19



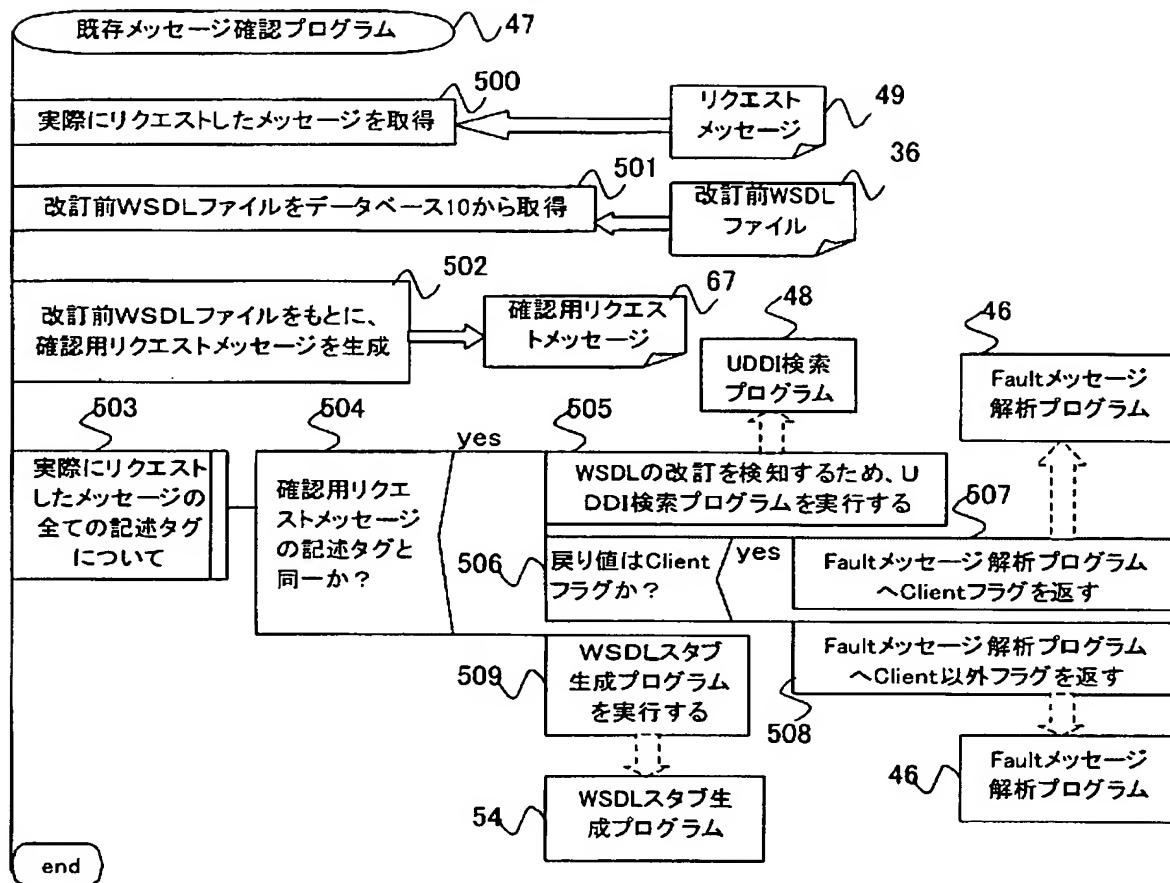
【図20】

四 20



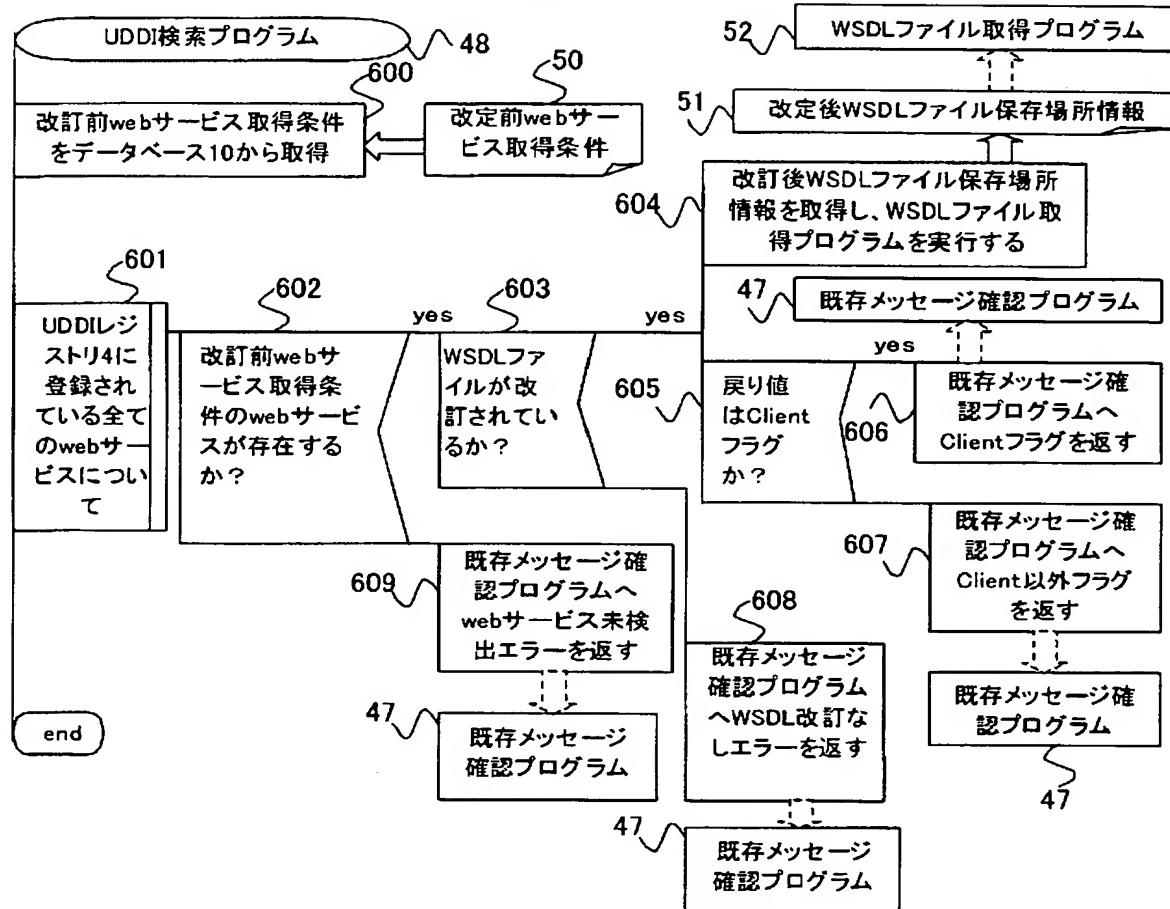
【図21】

図21



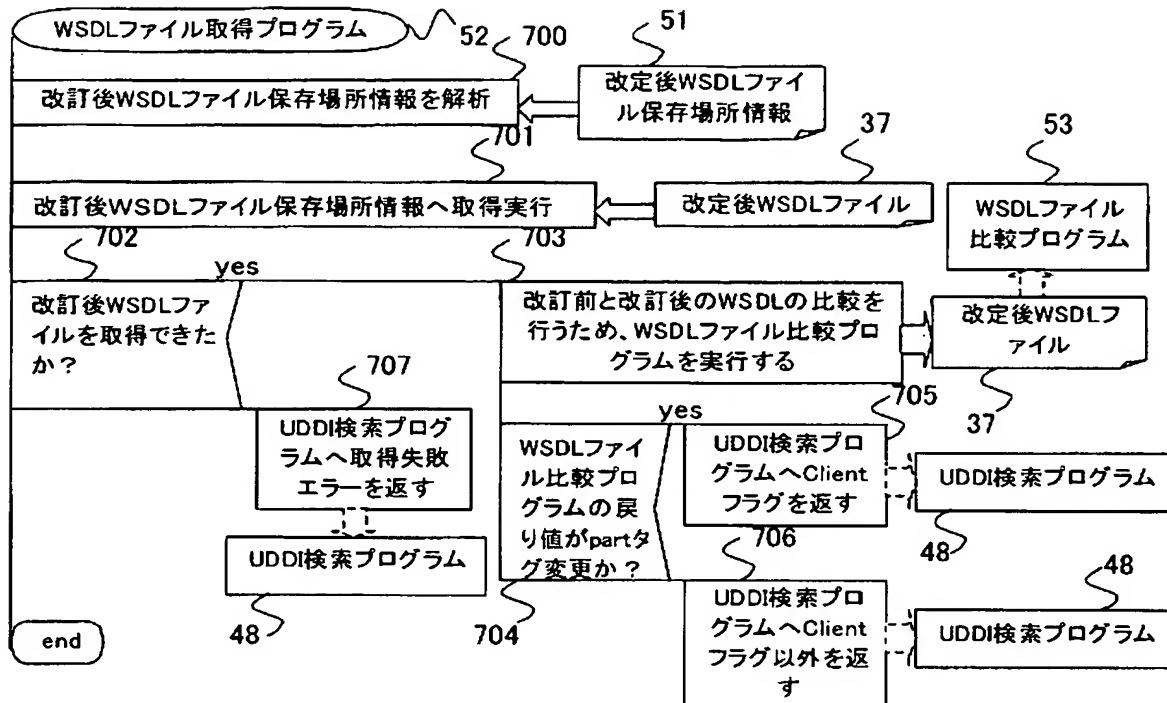
【図22】

図22



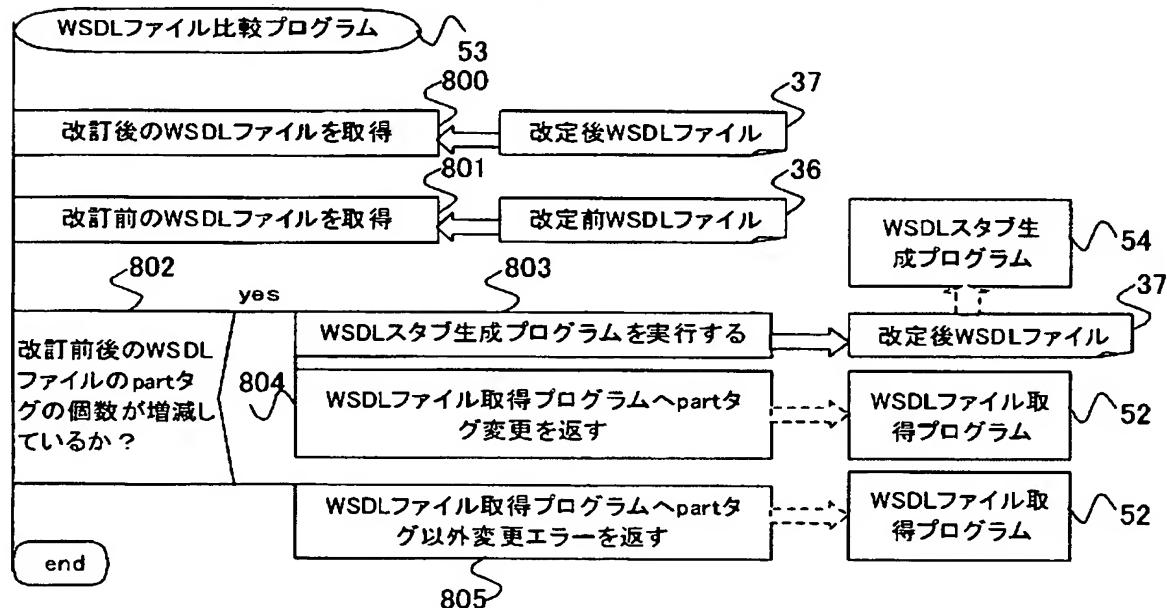
【図23】

図23



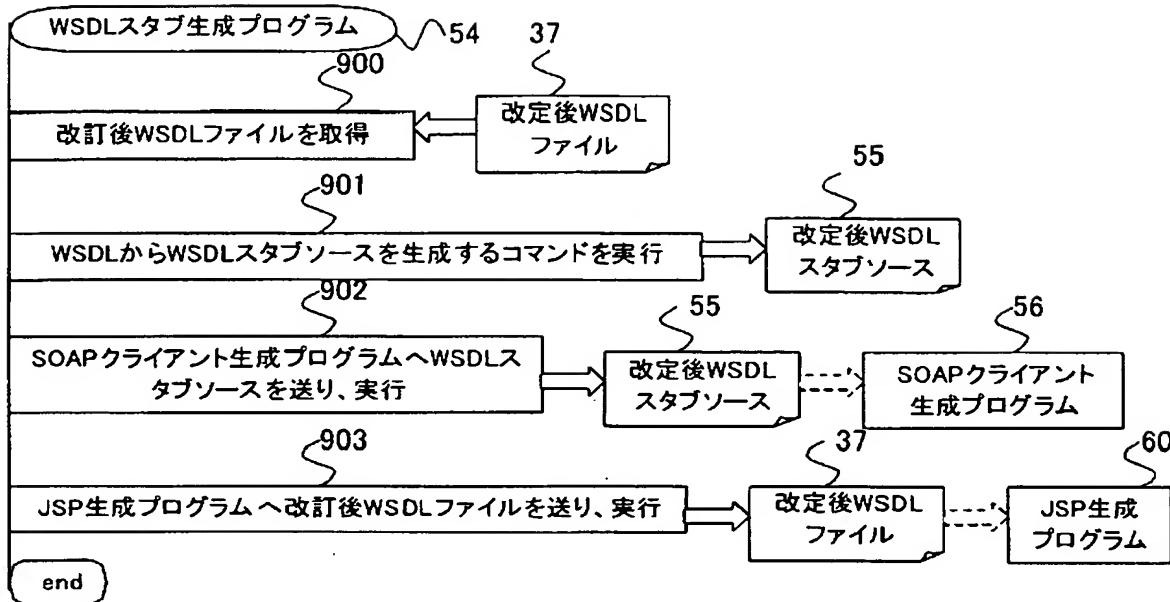
【図2-4】

图 24



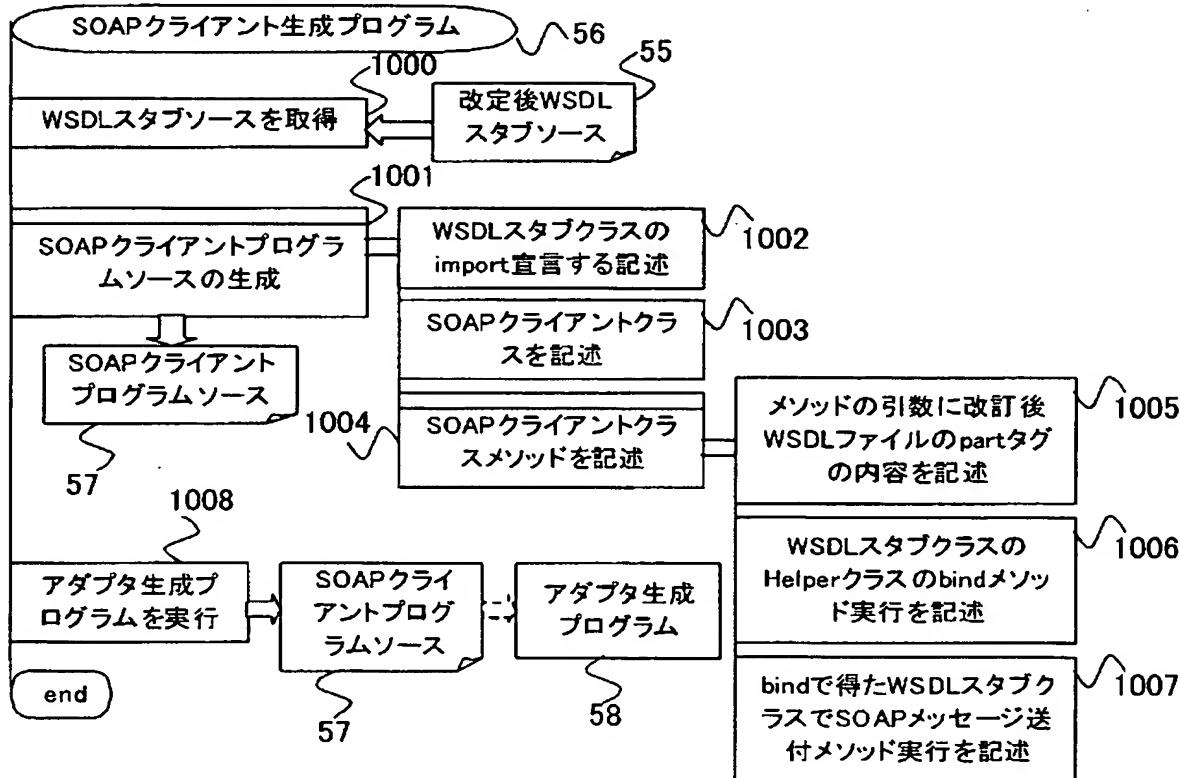
【図 25】

図 25

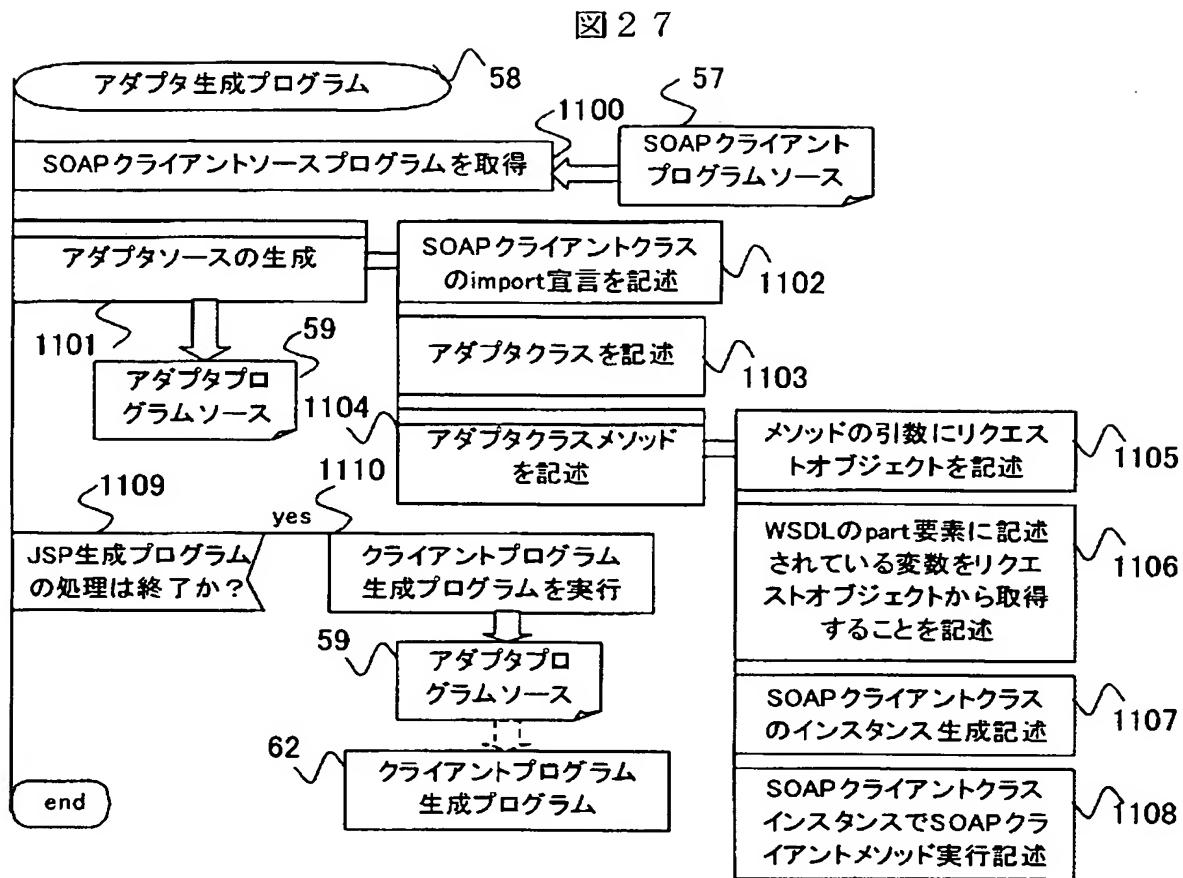


【図 26】

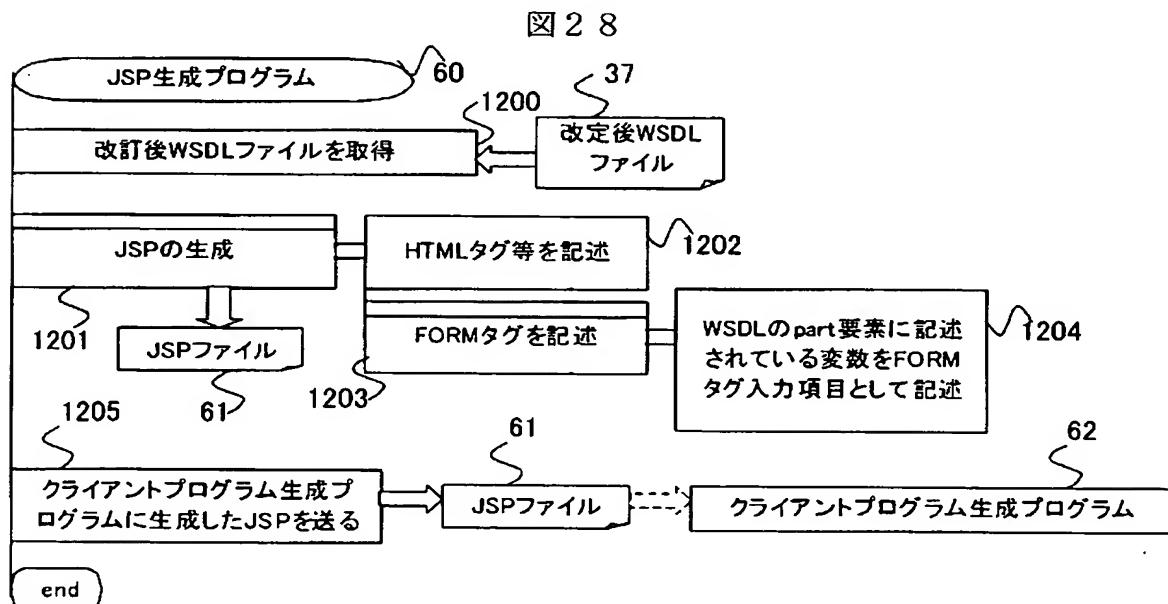
図 26



【図27】



【図28】



特願2003-057936

ページ： 19/E

出証特2003-3088775

【書類名】 要約書

【要約】

【課題】

実行中のクライアントプログラムについて、インターフェースの変更による修正、再配置、停止、再実行等の変更管理を動的に行うことを可能とする。

【解決手段】

クライアントプログラムが、サーバプログラムを提供している所定URLへアクセス要求を送付して、エラーが発生したとき、このエラーの発生原因を解析するためのプログラムを起動し、インターフェース定義情報の変更によるエラーか否かを判定し、インターフェース定義情報の変更によるエラーの場合は、上記クライアントプログラムの変更箇所を解析するプログラムを起動し、変更箇所の特定を行い、所定手続きに基づいて上記クライアントプログラムの上記変更箇所を変更して、上記クライアントプログラムを実行する。

【選択図】 図1

認定・付加情報

特許出願の番号 特願2003-057936
受付番号 50300353219
書類名 特許願
担当官 小野寺 光子 1721
作成日 平成15年 3月 7日

<認定情報・付加情報>

【提出日】 平成15年 3月 5日

次頁無

特願2003-057936

出願人履歴情報

識別番号 [000005108]

1. 変更年月日 1990年 8月31日

[変更理由] 新規登録

住所 東京都千代田区神田駿河台4丁目6番地
氏名 株式会社日立製作所